Constructing Bayesian finite mixture models by the EM algorithm

Petri Kontkanen, Petri Myllymäki, Henry Tirri
University of Helsinki[1]

NeuroCOLT Technical Report Series

NC-TR-97-003

February 1997[2]

Email:
Firstname.Lastname@cs.helsinki.fi

Produced as part of the ESPRIT Working Group
in Neural and Computational Learning,
NeuroCOLT 8556

NeuroCOLT Coordinating Partner

Royal Holloway
University of London

Department of Computer Science
Egham, Surrey TW20 0EX, England

For more information contact John Shawe-Taylor at the above address
or email neurocolt@dcs.rhbnc.ac.uk

[1]Complex Systems Computation Group (CoSCo), PO Box 26, Dept of Computer Science, FIN-00014 University of Helsinki, Finland

[2]Received December 3, 1996

NeuroCOLT Technical Report NC-TR-97-003.

Constructing Bayesian finite mixture models by the EM algorithm

Petri Kontkanen Petri Myllymäki Henry Tirri
Complex Systems Computation Group (CoSCo)
P.O.Box 26, Department of Computer Science
FIN-00014 University of Helsinki, Finland
URL: http://www.cs.Helsinki.FI/research/cosco/
Email: Firstname.Lastname@cs.Helsinki.FI

Abstract

In this paper we explore the use of finite mixture models for building decision support systems capable of sound probabilistic inference. Finite mixture models have many appealing properties: they are computationally efficient in the prediction (reasoning) phase, they are universal in the sense that they can approximate any problem domain distribution, and they can handle multimodality well. We present a formulation of the model construction problem in the Bayesian framework for finite mixture models, and describe how Bayesian inference is performed given such a model. The model construction problem can be seen as missing data estimation and we describe a realization of the Expectation-Maximization (EM) algorithm for finding good models. To prove the feasibility of our approach, we report crossvalidated empirical results on several publicly available classification problem datasets, and compare our results to corresponding results obtained by alternative techniques, such as neural networks and decision trees. The comparison is based on the best results reported in the literature on the datasets in question. It appears that using the theoretically sound Bayesian framework suggested here the other reported results can be outperformed with a relatively small effort.

Keywords: Finite mixture models, EM algorithm, Multinomial-Dirichlet model

1 Introduction

The purpose of our research is to explore approaches for building decision support systems capable of sound probabilistic inference. There are two principal approaches for constructing such a system: a model learning approach, where a (statistical) model of the problem domain distribution is constructed from a set of representative data samples, and a knowledge acquisition approach, where the knowledge of domain experts is coded in a suitable manner to form a model of the problem domain. In order to build systems capable of making good predictions, we adopt here the Bayesian approach [11], which offers a solid theoretical framework for combining both sources of information in the model construction process. An example of such an approach is given in [13] in the case of the Bayesian network model family.

Although Bayesian networks are an intuitively appealing, a theoretically sound family of probabilistic models, in practice there are some serious computational problems involved when working with general Bayesian network structures. First of all, learning Bayesian networks from data is a computationally very difficult problem, as the number of possible network structures grows superexponentially with respect to the number of variables. Secondly, even if we succeed in finding a good Bayesian network model, it is well known that probabilistic reasoning used for prediction in multi-connected Bayesian networks is an NP-hard problem [6, 7, 29], and hence very probably not computationally feasible in general. Consequently, most existing Bayesian network systems performing probabilistic reasoning first transform a given (learned) multi-connected BN structure to a singly-connected network by variable clustering or conditioning methods, and then use the existing polynomial-time algorithms (see e.g. [27, 26]) for solving probabilistic reasoning tasks. However, it is clear that as the basic problem is NP-hard, any transformation method may in the worst case take exponential time, or result in large conditional probability tables which cause the polynomial-time reasoning algorithms of the resulting singly-connected network to work in exponential time with respect to the original, multi-connected network.

Consequently dealing with statistical information with Bayesian networks we have to solve two difficult problems — the network construction problem and the the network transformation problem — before we are able to use the resulting models for prediction purposes. Therefore we adopt here an alternative approach, and restrict ourselves to a simpler, computationally efficient class of probabilistic models, the family of *finite mixture models* [10, 33]. By restricting ourselves to finite mixture models we gain the benefit of efficient probabilistic reasoning phase, and just have to concern ourselves with the complexity of the model construction problem.

A finite mixture model for a set of random variables is a weighted sum of a relatively small number of independent mixing distributions (for a more formal description see Section 2). As already mentioned, the main advantage of using finite mixture models lies in the fact that the computations for probabilistic reasoning can be implemented as a single pass computation as we will demonstrate in Section 3. The model also conforms interestingly to the intuitively appealing *memory-based reasoning* paradigm (see e.g., [17]), and it can be seen to offer a Bayesian solution to the case matching and case adaptation problems in such domains [20].

When compared to single parametric distribution methods and nonparametric methods (such as kernel density estimators [30]), finite mixture models have many appealing properties. Firstly, in natural domains the distribution to be modeled is often so complicated that probabilistic modeling through single parametric distribution (e.g., normal, multinomial, Poisson) does not lead to good approximations. It seems that distribution multimodality is more a rule than an exception, which naturally leads to serious problems for classical models. Finite mixtures have a natural means to model multimodality by placing a component distribution around each mode. Secondly, finite mixture models are universal in the sense that they can approximate any distribution arbitrarily close as long as a sufficient number of component densities is used. Thirdly, many non-parametric approaches depend on the sample size. So if the training sample is large, non-parametric techniques are inefficient in terms of computation time and space. Finite mixtures instead compress the information given by the sample at two levels: first by grouping similar data instances to a same cluster, and second by modeling a set of cluster instances by a single mixing distribution.

In the framework of the theory of Bayesian networks, a finite mixture model can be seen as a simple tree, where the root of the tree represents a latent (hidden) variable, corresponding to the mixing distributions, and the leaves represent the actual random variables of the problem domain (see [25]). Therefore, instead of searching over the exponentially many Bayesian network structures, we have a fixed model structure. However, in order to find good models, we have to search over the missing values of the unobserved latent variable in the dataset. In theory there is an exponential number of possible value combinations for this latent variable, but fortunately efficient algorithms for estimating this type of missing data exist. In Section 4 we formulate the model construction problem in our Bayesian framework, and in Section 5 show how to apply the Expectation-Maximation (EM) algorithm [9] in the learning process.

As our main motivation is building tools for decision making, we are more interested in the predictive capability of our models than their semantic interpretation. Consequently, although our approach akin to the AutoClass system [4], our application areas are somewhat different: the industrial applications we are currently working on are mainly focused on nonlinear regression, classification and configuration problems, whereas the AutoClass system has been successfully used for data mining problems, such as LandSat data clustering [5]. However, it should also be noted that for acceptable predictive performance, it seems reasonable to assume that the systems have to be based on good models of the problem domain, and hence very probably have also a structure which reflects reality well. In addition, although finite mixtures are semantically not quite as intuitive as Bayesian networks, in many cases the domain experts seem to be able to express their expert knowledge very easily by using prototypical examples or distributions, which can then be coded as mixing distributions in our finite mixture framework.

When constructing finite mixture models from data with data mining applications in mind, it should be noticed that in the Bayesian framework there does not exist a single "true" finite mixture model which can be identified from data, but several equally good models [12]. As our main focus is in prediction and not in data mining, this nonidentifiability is not a major concern for us: from a set of equally well performing models, we just pick randomly one that we use for making predictions. Theoretically the correct Bayesian approach would be to use the sum of outcomes of all the possible different models, weighted by their probability, i.e., in our case a "mixture of all the mixtures". This is clearly not feasible in practice, but like successful demonstrations with Bayesian networks suggest, to achieve a reasonable accuracy it is often enough to just use a single, maximum a posterior probability model. Our experimental results reported in Section 6 support this observation as they are obtained by using only a single finite mixture model for each problem domain.

2 The finite mixture model

We model here the problem domain by m discrete random variables $X_1, \ldots, X_m$ (continuous variables are at this point discretized by quantization). A *data instantiation* $\vec{d}$ is a vector in which all the variables X_i have been assigned a value, $\vec{d} = (X_1 = x_1, \ldots, X_m = x_m)$, where $x_i \in \{x_{i1}, \ldots, x_{in_i}\}$. A *random sample* $D = (\vec{d}_1, \ldots, \vec{d}_N)$ is a set of N i.i.d. (independent and identically distributed) data instantiations, where each $\vec{d}_j$ is sampled from $\mathcal{P}$, the joint distribution of the variables $(X_1, \ldots, X_m)$.

We assume that the data D is generated by K different mechanisms, which all can have their own distribution. Furthermore, we assume that each data vector originates from exactly one of these mechanisms. Thus the instantiation space is divided into K local regions called *clusters*, each of which consists of the data vectors generated by the corresponding mechanism. An appropriate statistical model for the case is a *finite mixture* ([10], [33]). In finite mixture models the problem domain probability distribution is approximated as a weighted sum of mixture distributions:

$$P(\vec{d}) = \sum_{k=1}^{K} P(Y = y_k) P(\vec{d}|Y = y_k), \tag{1}$$

where the values of the discrete *clustering random variable* Y correspond to the separate clusters of the instantiation space, and each mixture distribution $P(\vec{d}|Y = y_k)$ models one data producing mechanism. Moreover, we assume that the problem domain data is tightly clustered so that the clusters can actually be regarded as points in the instantiation space, and data vectors belonging to the same cluster just as noisy versions of that (unknown) point. It follows that we can assume the variables X_i inside each cluster to be independent and thus (1) becomes

$$P(\vec{d}) = P(X_1 = x_1, \ldots, X_m = x_m) = \sum_{k=1}^{K} P(Y = y_k) \prod_{i=1}^{m} P(X_i = x_i|Y = y_k).$$

Furthermore, in our model both the cluster distribution $P(Y)$ and the intra-class conditional distributions $P(X_i|Y = y_k)$ are multinomial. Thus a finite mixture model can be defined by first fixing K, the *model class* (the number of the mixing distributions), and then by determining the values of the model parameters $\Theta = (\alpha, \Phi), \Theta \in \Omega$, where

$$\alpha = (\alpha_1, \ldots, \alpha_K) \text{ and } \Phi = (\Phi_{11}, \ldots, \Phi_{1m}, \ldots, \Phi_{K1}, \ldots, \Phi_{Km}),$$

with the denotations

$$\alpha_k = P(Y = y_k), \Phi_{ki} = (\phi_{ki1}, \ldots, \phi_{kin_i}), \text{ where } \phi_{kil} = P(X_i = x_{il} | Y = y_k).$$

A detailed description of how to determine K and Θ, given a data sample D, can be found in Section 4.

3 Bayesian inference by finite mixture models

Given a finite mixture model, it is possible to solve various probabilistic reasoning tasks in a computationally efficient manner. The Bayesian *predictive inference* (see e.g. [2]) aims at predicting future (yet unobserved) quantities by means of already observed quantities. More precisely, if $\vec{d}$ is an unobserved data vector, its predictive distribution given observed data D is defined as

$$P(\vec{d}|D) = \int_{\Omega} P(\vec{d}|D, \Theta) P(\Theta|D) d\Theta. \tag{2}$$

So the predictive distribution can be considered as the expected distribution of $\vec{d}$ over all possible parameter settings (models). From (2) we see that the correct Bayesian procedure for making predictions would be to use all possible models by weighting them by their posterior density. However, we currently approximate the predictive distribution by using only a single parameter set $\hat{\Theta}$ which maximizes the density $P(\Theta|D)$, i.e. $\hat{\Theta}$ is a *maximum a posterior* (MAP) estimate of Θ. By setting the probability $P(\hat{\Theta}|D)$ to unity, the approximative predictive distribution becomes simply

$$P(\vec{d}|D) = P(\vec{d}|D, \hat{\Theta}) = P(\vec{d}|\hat{\Theta}), \tag{3}$$

where the last equality follows from the fact that $\vec{d}$ and D are conditionally independent given $\hat{\Theta}$.

With practical applications in mind, we are especially interested in finding the *conditional predictive distribution* of each uninstantiated X_i, given that some of the variables are instantiated, i.e. assigned a fixed value. Let $\mathcal{I} = \{i_1, \ldots, i_t\}$ be the indices of the instantiated variables, and let $\mathcal{X} = \{X_{i_s} = x_{i_s l_s}, s = 1, \ldots, t\}$ denote the corresponding assignments. Now we want to determine the distribution

$$P(X_i | D, \mathcal{X}) = \int_{\Omega} P(X_i | \Theta, \mathcal{X}) P(\Theta|D) d\Theta \tag{4}$$

for all $i \notin \mathcal{I}$. By using the approximation (3), (4) becomes

$$P(X_i = x_{il}|D,\mathcal{X}) = P(X_i = x_{il}|\hat{\Theta},\mathcal{X}) = \frac{P(X_i = x_{il},\mathcal{X}|\hat{\Theta})}{P(\mathcal{X}|\hat{\Theta})}$$

$$= \frac{\sum_{k=1}^{K}\left(P(Y = y_k|\hat{\Theta})P(X_i = x_{il}|Y = y_k,\hat{\Theta})\prod_{s=1}^{t}P(X_{i_s} = x_{i_s l_s}|Y = y_k,\hat{\Theta})\right)}{\sum_{k=1}^{K}\left(P(Y = y_k|\hat{\Theta})\prod_{s=1}^{t}P(X_{i_s} = x_{i_s l_s}|Y = y_k,\hat{\Theta})\right)}$$

$$= \frac{\sum_{k=1}^{K}\left(\hat{\alpha}_k\hat{\phi}_{kil}\prod_{s=1}^{t}\hat{\phi}_{ki_s l_s}\right)}{\sum_{k=1}^{K}\left(\hat{\alpha}_k\prod_{s=1}^{t}\hat{\phi}_{ki_s l_s}\right)}.$$

The conditional predictive distribution of X_i can clearly be calculated in time $\mathcal{O}(Ktn_i)$, where K is the number of clusters, t the number of instantiated variables and n_i the number of values of X_i. K is usually small compared to the sample size N, and thus the prediction computation can be performed very efficiently[1].

The predictive distributions can be used for classification and regression tasks. In classification problems, we have a special class variable X_c which is used for classifying data. We fix all the other variables and predict the value of X_c by using its conditional predictive distribution. With the simple 0/1-loss function, the predicted value is usually simply taken as the most probable value of the distribution in question, but alternative schemes emerge when using different loss functions. In more general regression tasks, we have more than one variable for which we want to compute the predictive distribution, given that the values of the other variables are instantiated in advance. Alternatively, finite mixture models can also be used for finding the maximal configuration (the most probable value assignment combination) for all the uninstantiated variables, given the values of the instantiated variables.

4 Learning finite mixture models from data

Let $D = (\vec{d}_1,\ldots,\vec{d}_N)$ be a random sample of N data instantiations, sampled from the problem domain distribution $\mathcal{P}$. By learning we mean here the problem of constructing a single finite mixture model $M_K(\Theta)$ which represents $\mathcal{P}$ as accurately as possible. The learning process can be divided into two separate phases: in the first phase we wish to determine the optimal value for K, the number of mixing distributions (the model class), and in the second phase we wish to find MAP parameter values $\hat{\Theta}$ for the chosen model class.

4.1 Determining the model class

In the Bayesian framework, the optimal number of mixing distributions (clusters) can be determined by evaluating the posterior probability for each model class $\mathcal{M}_K$

[1] If massively parallel hardware is available, the computations can be made even faster since the algorithms can be parallelized easily [24, 23].

given the data:

$$P(\mathcal{M}_K|D) \propto P(D|\mathcal{M}_K)P(\mathcal{M}_K), K = 1, \ldots, N,$$

where the normalizing constant $P(D)$ can be omitted since we only need to compare different model classes. The number of clusters can safely be assumed to be bounded by N, since otherwise the sample size is clearly too small for the learning problem in question.

Assuming equal priors for the model classes, they can be ranked by evaluating the *evidence* $P(D|\mathcal{M}_K)$ for each model class,

$$P(D|\mathcal{M}_K) = \int_\Omega P(D|\Theta, \mathcal{M}_K)P(\Theta|\mathcal{M}_K)\, d\Theta. \tag{5}$$

This integral is very hard to evaluate since the dimension of the parameter space is usually very large ($K(1 + \sum_{i=1}^m n_i)$), but the evidence can be approximated by using e.g. Laplace's method [16]. Many information-theoretic measures, such as stochastic complexity [28], can also be seen as approximations of this integral. However, in the experimental results presented in Section 6 we did not use the evidence for automated model class selection, since we found the public domain datasets used too small for reasonable accuracy approximations. Instead, in the search process the model classes were selected by manually searching the model class space for good values of K, and comparing different model classes by their crossvalidated performance on the data.

4.2 Determining the model class parameters

Let us assume that the model class $\mathcal{M}_K$ is fixed. In the Bayesian approach, the goal is to find the MAP (maximum a posteriori) estimate of parameters by maximizing the posterior density $P(\Theta|D)$,

$$P(\Theta|D) = \frac{P(D|\Theta)P(\Theta)}{\int_\Omega P(D|\Theta)P(\Theta)\, d\Theta},$$

where the normalizing factor is the evidence term (5) discussed in the previous section.

We assume that the prior distributions of the parameters are from the family of Dirichlet densities,

$$(\alpha_1, \ldots, \alpha_K) \sim \mathrm{Di}\,(\mu_1, \ldots, \mu_K)\,, \text{ and} \tag{6}$$

$$(\phi_{ki1}, \ldots, \phi_{kin_i}) \sim \mathrm{Di}\,(\sigma_{ki1}, \ldots, \sigma_{kin_i})\,, k = 1, \ldots, K, i = 1, \ldots, m. \tag{7}$$

Assuming that the parameter vectors α and Φ_{ki} are independent, the joint prior distribution of all the parameters is

$$\mathrm{Di}\,(\mu_1, \ldots, \mu_K) \prod_{k=1}^K \prod_{i=1}^m \mathrm{Di}\,(\sigma_{ki1}, \ldots, \sigma_{kin_i}) \tag{8}$$

with density

$$P(\Theta) = \Gamma\left(\sum_{k=1}^{K}\mu_k\right)\prod_{k=1}^{K}\left(\frac{\alpha_k^{\mu_k-1}}{\Gamma(\mu_k)}\right)\prod_{k=1}^{K}\prod_{i=1}^{m}\left(\Gamma(\sum_{l=1}^{n_i}\sigma_{kil})\prod_{l=1}^{n_i}\frac{\phi_{kil}^{\sigma_{kil}-1}}{\Gamma(\sigma_{kil})}\right).$$

The family of Dirichlet densities is *conjugate* (see e.g. [8]) to the family of multinomials, i.e. the functional form of parameter distribution remains invariant in the prior-to-posterior transformation, thus the choice of Dirichlet priors is justified.

The likelihood $P(D|\Theta)$ has two different forms depending on whether we know to which cluster each data vector belongs to or not. Let us first consider the positive case, i.e. *supervised learning.* The likelihood term can now be conveniently represented by introducing the *cluster indicator variables* $Z = (z_1, \ldots, z_N)$, $z_j = (z_{j1}, \ldots, z_{jK})$, where

$$z_{jk} = \begin{cases} 1, & \text{if } \vec{d_j} \in y_k, \\ 0, & \text{otherwise.} \end{cases}$$

It follows that

$$P(D|\Theta, \mathcal{M}_K) = \prod_{k=1}^{K}\left(\alpha_k^{h_k}\prod_{i=1}^{m}\prod_{l=1}^{n_i}\phi_{kil}^{f_{kil}}\right),$$

where $h_k = \sum_{j=1}^{N} z_{jk}$ is the number of instantiations in cluster k, and $f_{kil} = \sum_{j=1}^{N} z_{jk}v_{jil}$ is the number of instantiations in cluster k with variable X_i having value x_{il}, where

$$v_{jil} = \begin{cases} 1, & \text{if } d_{ji} = x_{il}, \\ 0, & \text{otherwise.} \end{cases}$$

From the conjugacy property it follows that the posterior distribution is also a product of Dirichlet's and is given by

$$\mathrm{Di}\left(\mu_1 + h_1, \ldots, \mu_K + h_K\right)\prod_{k=1}^{K}\prod_{i=1}^{m}\mathrm{Di}\left(\sigma_{ki1} + f_{ki1}, \ldots, \sigma_{kin_i} + f_{kin_i}\right)$$

with the corresponding density $P(\Theta|D)$ being

$$\Gamma\left(\sum_{k=1}^{K}(\mu_k + h_k)\right)\prod_{k=1}^{K}\left(\frac{\alpha_k^{\mu_k+h_k-1}}{\Gamma(\mu_k + h_k)}\right)\prod_{k=1}^{K}\prod_{i=1}^{m}\left(\Gamma\left(\sum_{l=1}^{n_i}(\sigma_{kil} + f_{kil})\right)\prod_{l=1}^{n_i}\frac{\phi_{kil}^{\sigma_{kil}+f_{kil}-1}}{\Gamma(\sigma_{kil} + f_{kil})}\right). \quad (9)$$

The MAP estimate $\Theta = \hat{\Theta}$ is the mode of (9), and since the parameters α_k and ϕ_{kil} appear disjointly, the mode can be determined by finding the modes of the individual Dirichlet's and is obtained by setting

$$\hat{\alpha}_k = \frac{h_k + \mu_k - 1}{N + \sum_{k=1}^{K}\mu_k - K}, \quad \hat{\phi}_{kil} = \frac{f_{kil} + \sigma_{kil} - 1}{h_k + \sum_{l=1}^{n_i}\sigma_{kil} - n_i}. \quad (10)$$

Assuming that the values of the cluster indicators are part of the training data D results in the simple naive Bayes model (see e.g. [15]). If, however, we do not consider the values of cluster indicators to be known, the learning problem becomes much more complex. In this *unsupervised learning* case we regard z_j's as random variables and assume that Z is a random sample of size N from a multinomial distribution with parameters α. The joint likelihood of D and Z is

$$P(D, Z|\Theta) = \prod_{j=1}^{N} \prod_{k=1}^{K} \left(\alpha_k \prod_{i=1}^{m} \phi_{kid_{ji}} \right)^{z_{jk}}, \tag{11}$$

and the marginal likelihood of D is obtained by summing over Z corresponding to all possible clusterings of data into K clusters. These clusterings are, however, exponential (K^N) in number, so exact MAP estimate cannot be found, and we are forced to use numerical approximation methods. The usual approach is to regard Z as missing data and estimate it iteratively [21]. In the next section, we show how to use the *Expectation-Maximization (EM)* algorithm [9] for solving this problem.

5 Estimating the MAP parameters by the EM algorithm

In the sequel, we refer to D as incomplete (or observed) data, and to (D, Z) as complete data, where Z denotes the values of the cluster indicators as in section 4.2. Perhaps the most commonly algorithm for estimating missing data is the EM algorithm. In the following, we first describe shortly the general EM framework, and show then how to apply the method for learning finite mixture models with uniform and non-uniform priors.

5.1 The EM algorithm

The EM algorithm consists of two steps: Expectation(E) and Maximization(M). In E step, the expected posterior density corresponding to the complete data given the incomplete data and the current parameter estimates is determined. In M step, the parameter values are updated in such a way that the obtained expected posterior is maximized. The intuition behind the algorithm is that we would like to maximize the complete data posterior but since we do not know it, we maximize its expectation instead. The EM algorithm is as follows:

1. Initialize parameters randomly. Set $t = 0$.
2. E-step: Determine $Q(\Theta, \Theta^t) = E[\log P(\Theta|D, Z)|D, \Theta^t]$.
3. M-step: Set $\Theta^{t+1} = \arg\max_{\Theta}\{Q(\Theta, \Theta^t)|\Theta \in \Omega)\}$, where Θ^t are the current parameter estimates in time step t.

4. Iterate until convergence.

The main disadvantage of the EM algorithm is its slow theoretical convergence rate, which is shown to be linear [9]. The convergence is especially slow near the mode(s) of the function being maximized. Nevertheless, the convergence of EM is monotonic and the algorithm is assured to converge to a local optimum. Although the two main competitors of EM, the Method of Scoring and Newton-Rapshon (see [33]), have a quadratic convergence rate, the methods lack the nice monotonic convergence property (and may not converge at all), and are also usually more difficult to apply than EM is. Moreover, according to our experiments reported in Section 6, in practice the EM algorithm seems to converge very rapidly to a reasonable approximation of the local optimum. This allows us to search for the global optimum by iteratively running EM with randomly chosen initializations of the parameters. Usually a relatively small number of iterations was sufficient to get good results (in terms of the crossvalidation error).

5.2 Implementation of EM assuming uniform priors

Let us assume that the prior distribution $P(\Theta)$ is uniform, i.e. $\{\mu_k = 1; \sigma_{kil} = 1; k = 1, \ldots, K, i = 1, \ldots, m, l = 1, \ldots, n_i\}$. As a result, the problem clearly reduces to maximum likelihood estimation, that is, in E-step, we have to find $E[\log P(D, Z|\Theta)|D, \Theta^t]$, the conditional expected complete data log-likelihood. From (11) we see that it is

$$E[\sum_{j=1}^{N}\sum_{k=1}^{K} z_{jk}(\log \alpha_k + \sum_{i=1}^{m} \log \phi_{kid_{ji}})|D, \Theta^t]. \tag{12}$$

By defining (vector) functions

$$U(\alpha) = (\log \alpha_1, \ldots, \log \alpha_K)^T \text{ and } V(\Phi) = (\sum_{i=1}^{m} \log \phi_{1id_{ji}}, \ldots, \sum_{i=1}^{m} \log \phi_{Kid_{ji}})^T,$$

and using the linearity of expectation, (12) becomes

$$\sum_{j=1}^{N} E[{z_j}^T|D, \Theta^t]U(\alpha) + \sum_{j=1}^{N} E[{z_j}^T|D, \Theta^t]V(\Phi).$$

Therefore, we only need the conditional expectations of the cluster indicators,

$$\begin{aligned} E[z_{jk}|D, \Theta^t] &= P(z_{jk} = 1|D, \Theta^t) = P(z_{jk} = 1|\vec{d_j}, \Theta^t) \\ &= \frac{P(z_{jk} = 1|\Theta^t)P(\vec{d_j}|z_{jk} = 1, \Theta^t)}{P(\vec{d_j}|\Theta^t)} = \frac{\alpha_k^t \prod_{i=1}^m \phi_{kid_{ji}}^t}{\sum_{k'=1}^K \left(\alpha_{k'}^t \prod_{i=1}^m \phi_{k'id_{ji}}^t\right)}, \end{aligned}$$

which are denoted from now on by w_{jk}. This completes the E-step.

In the M-step, we must find Θ^{t+1} such that

$$\sum_{j=1}^{N}\sum_{k=1}^{K} w_{jk}\log\alpha_k + \sum_{j=1}^{N}\sum_{k=1}^{K} w_{jk}\sum_{i=1}^{m}\log\phi_{kid_{ji}} \tag{13}$$

is maximized under the constraints

$$\left\{\sum_{k=1}^{K}\alpha_k = 1, \sum_{l=1}^{n_i}\phi_{kil} = 1; k = 1,\ldots,K, i = 1,\ldots,m\right\}.$$

Parameters α and Φ appear disjointly in (13) (and in their constraints), so they can be maximized separately. An easy application of standard constraint optimization method, the method of Lagrange multipliers, gives

$$\alpha_k^{t+1} = \frac{\sum_{j=1}^{N} w_{jk}}{N}, \quad \phi_{kil}^{t+1} = \frac{\sum_{j=1}^{N} w_{jk}v_{jil}}{\sum_{j=1} w_{jk}}. \tag{14}$$

It is easy to see that actually

$$\sum_{j=1}^{N} w_{jk} = \bar{h}_k = E[h_k|D,\Theta^t], \quad \sum_{j=1}^{N} w_{jk}v_{jil} = \bar{f}_{kil} = E[f_{kil}|D,\Theta^t]. \tag{15}$$

This result is intuitively pleasant in the sense that the M-step parallels the maximization phase of the supervised case (10) with h_k's and f_{kil}'s replaced by their expectations. This is not a coincidence, since the h_k's and f_{kil}'s are actually the sufficient statistics of the model, and an alternative formulation of the EM can be derived by maximizing only the expected sufficient statistics instead of the whole expected posterior.

5.3 Implementation of EM with general priors

Now we relax the restriction of priors and assume that the prior distribution of Θ is as defined by (6) and (7). There seems to be no computationally efficient way to evaluate the function Q directly in this general case. The standard way to overcome this problem is to apply the Bayes theorem sequentially considering only one data vector at a time, and then approximating the resulting posterior in a suitable way. This procedure, called the Quasi-Bayes algorithm [33] or fractional updating [32], was used in our empirical tests in Section 6.

Let us first assume that the dataset consists of only one data vector $\vec{d_1}$. Now the function Q can be expressed as

$$\begin{aligned} Q(\Theta,\Theta^t) &= E[P(\Theta|D,Z)|D,\Theta^t] \\ &= \sum_{k=1}^{K}\Big(w_{1k}\mathrm{Di}\left(\mu_1+z_{11},\ldots,\mu_K+z_{1K}\right)\prod_{k'=1}^{K}\prod_{i=1}^{m}\mathrm{Di}\left(\sigma_{k'i1}+z_{1k}v_{1i1},\ldots,\sigma_{k'in_i}+z_{1k}v_{1in_i}\right)\Big). \end{aligned}$$

This can be approximated by replacing each z_{jk} by its expectation w_{jk}, resulting in an approximation of the function Q after one data vector:

$$\tilde{Q}(\Theta, \Theta^t) = \mathrm{Di}\left(\mu_1 + w_{11}, \ldots, \mu_K + w_{1K}\right) \prod_{k=1}^{K} \prod_{i=1}^{m} \mathrm{Di}\left(\sigma_{ki1} + w_{1k} v_{1i1}, \ldots, \sigma_{kin_i} + w_{1k} v_{1in_i}\right).$$

Continuing in this manner until the whole database is processed, we finally get

$$\tilde{Q}(\Theta, \Theta^t) = \mathrm{Di}\left(\mu_1 + \bar{h}_1, \ldots, \mu_K + \bar{h}_k\right) \prod_{k=1}^{K} \prod_{i=1}^{m} \mathrm{Di}\left(\sigma_{ki1} + \bar{f}_{ki1}, \ldots, \sigma_{kin_i} + \bar{f}_{kin_i}\right), \tag{16}$$

where the $\bar{h}_k$'s and $\bar{f}_{kil}$'s are as defined by (15). This approximation seems natural in the sense that by replacing unknown z_{jk}'s by their expectations we essentially divide each datavector between the clusters according to their probabilities of originating from each cluster. Furthermore, the additive nature of the Dirichlet distribution justifies the summing of these probabilities. By additive nature we mean here that when new data vector arrives, the parameters of Dirichlet distribution are updated by adding the contribution of the new data vector to the old value of each parameter. An analysis of the convergence of the Quasi-Bayes procedure and a comparison to other similar methods is given in [33].

In the M-step we have to find the mode of (16). The problem is essentially the same as in supervised case, thus from (10), we get

$$\alpha_k^{t+1} = \frac{\bar{h}_k + \mu_k - 1}{N + \sum_{k=1}^{K} \mu_k - K}, \quad \phi_{kil}^{t+1} = \frac{\bar{f}_{kil} + \sigma_{kil} - 1}{\bar{h}_k + \sum_{l=1}^{n_i} \sigma_{kil} - n_i}. \tag{17}$$

Note that in the case of uniform priors, the formulas in (17) reduce to those in (14).

6 Empirical results

In the previous sections we have presented an EM-based approach for finding a good mixture model, i.e., for estimating the MAP parameters for finite mixture models with both uniform and non-uniform priors. However, to be able to derive the resulting scheme, several approximations (although theoretically justified) had to be made. Therefore an interesting question is, how well the presented framework works in practice.

The approach described above has been implemented as part of a more general software package for probabilistic inference (CABINET). As already discussed in the introduction, we are motivated in finding models with good predictive performance. In addition we are interested in evaluating our approach against other popular method families such as neural networks or decision tree algorithms, as well as more traditional statistical approaches (e.g. linear discriminant analysis). Therefore the choice whether to use artificial or natural data had to made. Either selection has its advantages and disadvantages.

To perform an extensive set of experiments with artificial (generated) data would be a very time-consuming task, but would allow us to also calculate the statistical significance of the differences found. However, in such a comparative study a serious validity problem would be faced. Almost all of the algorithms proposed in the literature are sensitive to the parameter selection, which usually affects drastically the results achieved. Since we cannot have expertise in all the possible algorithms, the results would be severely hampered by our (in)competence in the method used. Therefore, instead of using artificial data we decided to test the prediction performance for natural datasets. We are well aware of the problems in using such "real" datasets, especially for comparison purposes: it is very difficult to identify underlying causes for performance differences, there is a high variance in the observed performance differences etc. The main advantage of using natural datasets is that they are produced without any knowledge of the particular procedures that they are tested on. In addition, although there is no way of telling how the results for a natural data set generalizes to other problems, we at least know that there are some domains where the results have practical relevance.

Below we report results from an ongoing extensive experimentation with the Bayesian finite mixture modeling method using publicly available datasets for classification problems. For comparison purposes, we have also collected performance results from the literature for alternative methods using the same datasets. We make no claims that the list of the results of alternative algorithms is exhaustive, however for each dataset we have included the best results we have found in the literature. The selection of datasets was done on the basis of their reported use, i.e., we have preferred datasets that have been used for testing many different methods over datasets with only isolated results. Many of the results are from the StatLog project [22], but we have also included more recent results. The descriptions of the datasets, the testing procedures used, and the best model classes (the number of mixtures) found for each data set are given in Table 1. The last column, the default value, denotes the success rate of a simple classifier, which classifies all the instances to the most common class.

It should be observed that with the exception of the DNA dataset, all our current results are crossvalidated, and when possible (for the StatLog datasets) we have used the same crossvalidation scheme as described in [22]. The same does not hold for many of the results for the other methods. In many cases the testing procedure either was not reported, or the best result with a single test set was given. The current performance results of all the algorithms (measured as classification success percentage) for individual datasets are presented as barcharts in Figures 1 and 2.

Even from these preliminary empirical results it can clearly be seen that the Bayesian finite mixture approach performs well not only when compared to traditional memory-based methods such as K-NN, IB3, or ALLOC80, but also with respect to decision tree methods and common neural network approaches such as backpropagation. The goodness of the results with respect to the memory based methods are not especially surprising as they can be seen as approximations to the

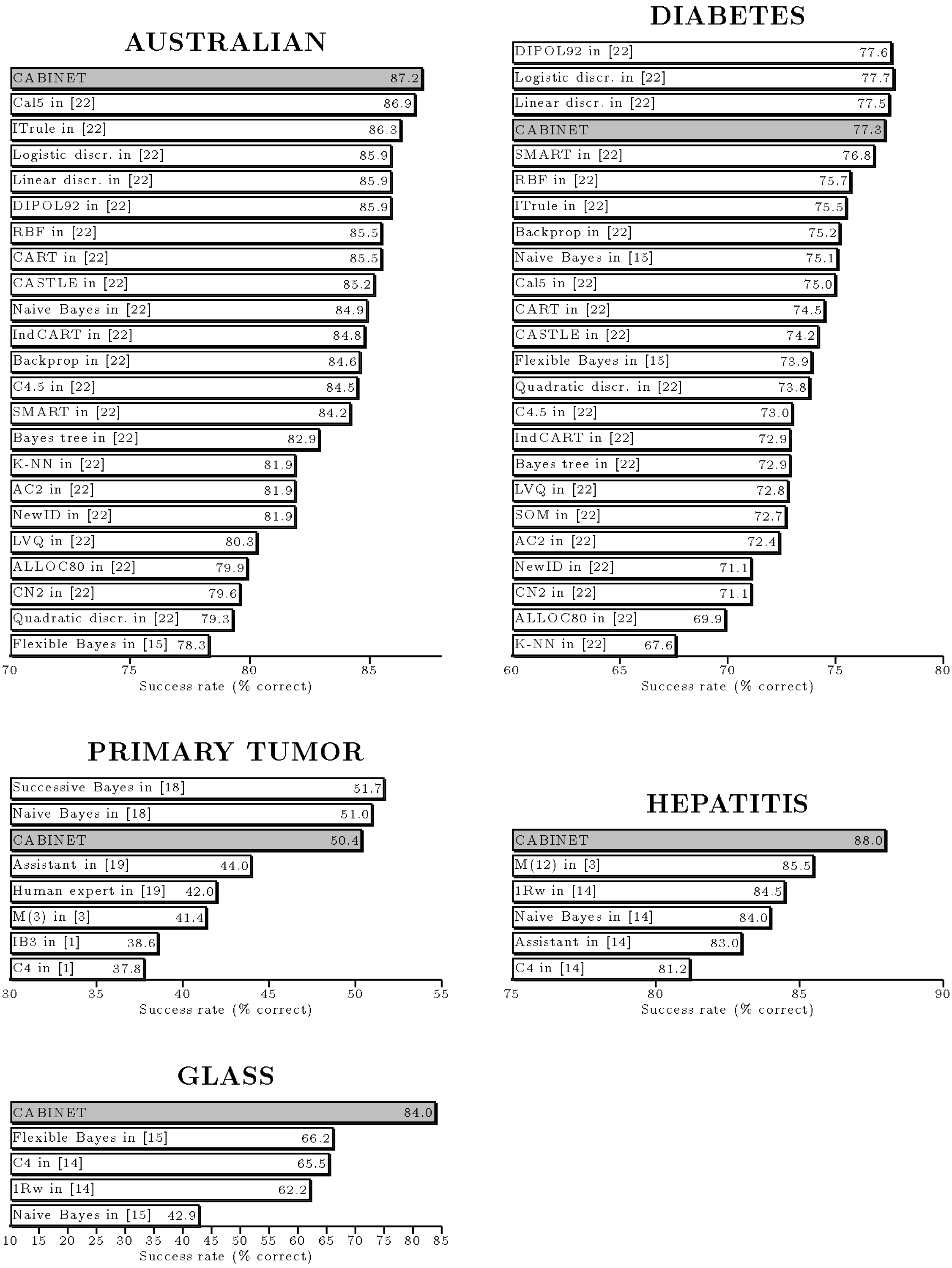

Figure 1: Experimental results on the Australian, Diabetes, Primary tumor, Hepatitis and Glass datasets

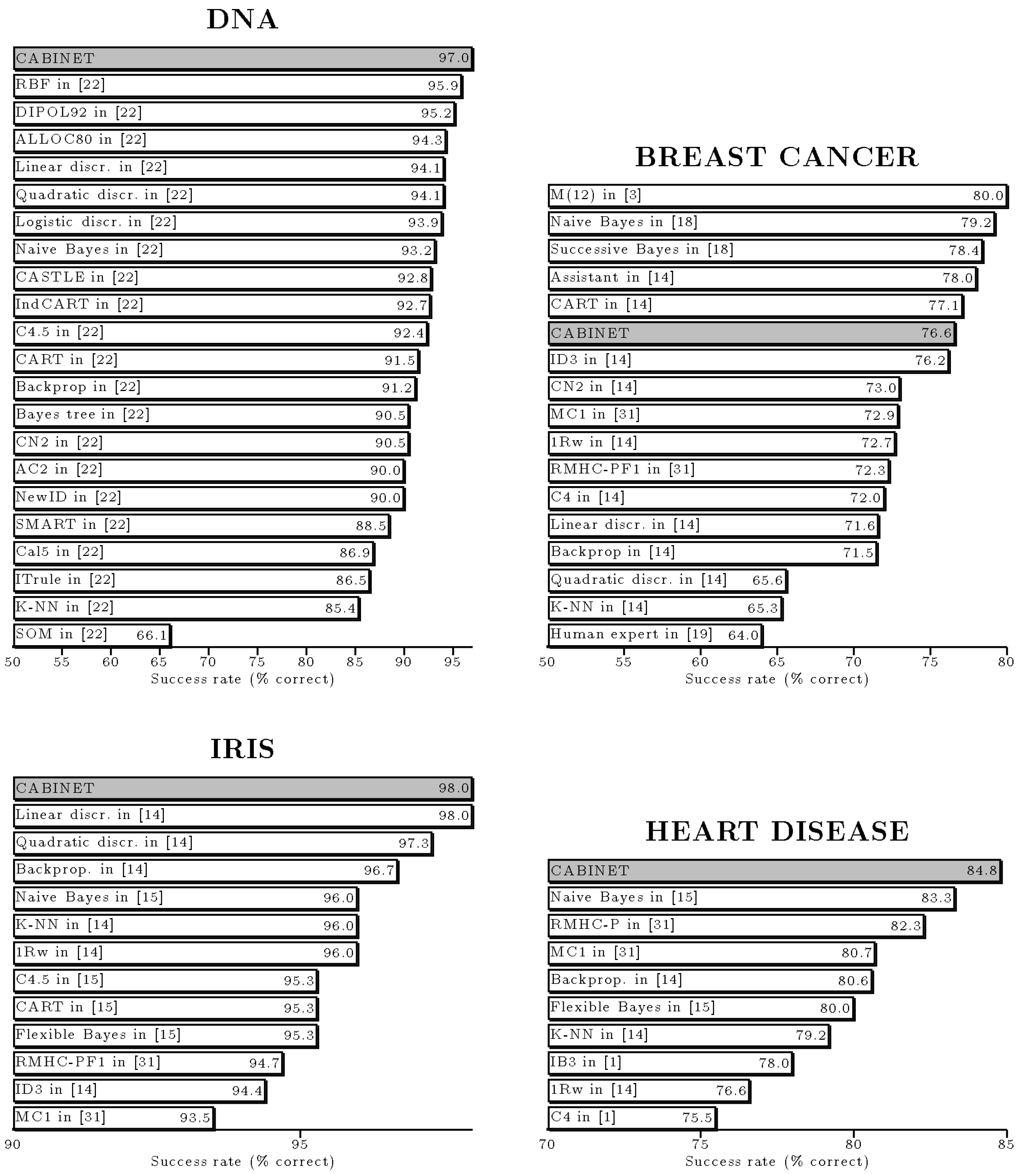

Figure 2: Experimental results on the DNA, Breast cancer, Iris and Heart disease databases

dataset	size	#attrs	#classes	#clusters	test method	default
Australian	690	15	2	20	10-fold CV	56.0
Breast cancer	286	10	2	21	11-fold CV	70.3
Diabetes	768	9	2	20	12-fold CV	65.0
DNA	3186	181	3	13	train&test	50.8
Glass	214	10	6	48	7-fold CV	40.7
Heart disease	270	14	2	9	9-fold CV	79.4
Hepatitis	150	20	2	6	5-fold CV	55.6
Iris	150	5	3	5	5-fold CV	33.3
Primary tumor	339	18	21	21	10-fold CV	24.8

Table 1: Description of the datasets and testing methods used in our experiments.

finite mixture approach presented here [20]. A more interesting observation is that the EM finds MAP estimates that outperform also all other Bayesian approaches present in the StatLog comparison including CASTLE, Naive Bayes, and Bayes tree algorithms. This can be interpreted as evidence for the common hypothesis that many real data distributions can be naturally modeled as a mixture of component distributions.

In order to summarize the results of different algorithms over many datasets we have collected these summaries in to Table 2. Currently this table summarizes only 10 algorithms for 4 datasets (Australian, Breast cancer, Diabetes, DNA) i.e., much less information than we have in the individual barcharts. This is due to the fact that this was the largest set of algorithms we could find that has been tested with the same data sets. By *performance index* we mean here the relative success percentage of a given method, when compared to the best method in the current classification task (e.g. the performance index of 90.0 of means that the method has achieved a success rate which is 90% of the success rate of the best method in the task in question). It is obvious that the results presented in the table have to be interpreted cautiously, since the datasets selected evidently do not reflect the variety present in the individual dataset results. For example the good performance of the linear discriminant analysis indicates that for many of the datasets the decision boundaries can at least be well approximated by hyperplanes. Nevertheless, the results confirm the observation already apparent from the results for the individual datasets, i.e., that the Bayesian finite mixture based approach offers the most consistent performance over the various datasets. We also want to point out that we have found the EM-based approach to be very effective in producing good models, and the results presented here have in many cases been found with very moderate computing resources.

algorithm	mean	variance	min	max
CABINET	99.11	1.93	96.72	100.00
Naive Bayes	97.59	2.15	96.08	100.00
Linear discr.	96.48	13.43	90.40	100.00
CART	96.46	1.99	94.33	98.05
Backprop.	94.59	7.70	90.28	97.03
C4 / C4.5	94.32	4.80	90.91	96.90
CN2	92.12	0.56	91.28	93.30
Quadratic discr.	91.50	29.94	82.83	97.01
K-NN	87.91	16.61	82.45	93.92
Default	72.31	216.84	52.37	88.76

Table 2: Performance indexes of some of the most commonly used methods for four datasets (Australian, Breast cancer, Diabetes, DNA) used in our experiments.

7 Conclusion

We have discussed the use of finite mixture models for building predictive probabilistic models. The presented Bayesian framework for finite mixture models is realized by using the Expectation-Maximization (EM) algorithm for finding good models. We also presented empirical results of the method's classification prediction performance for a set of public domain classification data sets, and compared the results to the performance of various machine learning and neural network methods. Although the experimentation is still ongoing, the current results clearly demonstrate that our Bayesian finite mixture approach is highly competitive, and offers a very consistent performance over different types of data.

One should observe that the derivation presented here is given for discrete data sets (but continuous model spaces). The Bayesian finite mixture approach extends also to the case where attributes are real-valued. However, some limited experimentation we performed indicated that moving from discrete to continuous values does not necessarily improve the prediction performance of the model. This is due to the additional assumptions of the distribution form (e.g., modeling continuous attributes with independent Gaussian distributions with appropriate conjugate priors) that are required, as well as to the fact that eventually when computations are performed, only limited numerical accuracy is available. We are currently investigating these issues.

Acknowledgments This research was supported by the Technology Development Center (TEKES). The primary tumor domain and the breast cancer domain were obtained from the University Medical Centre, Institute of Oncology, Ljubljana, Yugoslavia. Thanks go to M. Zwitter and M. Soklič for providing the data.

References

[1] D. Aha, D. Kibler, and M. Albert. Instance-based learning algorithms. *Machine Learning*, 6:37–66, 1991.

[2] J.M. Bernardo and A.F.M Smith. *Bayesian theory*. John Wiley, 1994.

[3] B. Cestnik and I. Bratko. On estimating probabilities in tree pruning. In Y. Kodratoff, editor, *Machine Learning EWSL-91*, pages 138–150. Springer-Verlag, 1991.

[4] P. Cheeseman, J. Kelly, M. Self, J. Stutz, W. Taylor, and D. Freeman. Autoclass: A Bayesian classification system. In *Proceedings of the Fifth International Conference on Machine Learning*, pages 54–64, Ann Arbor, June 1988.

[5] P. Cheeseman and J. Stutz. Bayesian classification (AutoClass): Theory and results. In U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, chapter 6. AAAI Press, Menlo Park, 1995.

[6] G.F. Cooper. The computational complexity of probabilistic inference using Bayesian belief networks. *Artificial Intelligence*, 42(2–3):393–405, March 1990.

[7] P. Dagum and M. Luby. Approximating probabilistic inference in Bayesian belief networks is NP-hard. *Artificial Intelligence*, 60:141–153, 1993.

[8] M.H. DeGroot. *Optimal statistical decisions*. McGraw-Hill, 1970.

[9] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38, 1977.

[10] B.S. Everitt and D.J. Hand. *Finite Mixture Distributions*. Chapman and Hall, London, 1981.

[11] A. Gelman, J. Carlin, H. Stern, and D. Rubin. *Bayesian Data Analysis*. Chapman & Hall, 1995.

[12] M. Gyllenberg, H.G. Gyllenberg, T. Koski, and J Schindler. Non-uniqueness of numerical taxonomic structures. *Binary*, 5:138–144, 1993.

[13] D. Heckerman, D. Geiger, and D.M. Chickering. Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20(3):197–243, September 1995.

[14] R.C. Holte. Very simple classification rules perform well on most commonly used datasets. *Machine Learning*, 11:63–91, 1993.

[15] G.H. John and P. Langley. Estimating continuous distributions in Bayesian classifiers. In P. Besnard and S. Hanks, editors, *Proceedings of the 11th Conference on Uncertainty in Artificial Intelligence*, pages 338–345. Morgan Kaufmann Publishers, 1995.

[16] R.E. Kass and A.E. Raftery. Bayes factors. Technical Report 254, Department of Statistics, University of Washington, 1994.

[17] H. Kitano. Challenges of massive parallelism. In *Proc. of IJCAI-93, the Thirteenth International Joint Conference on Artificial Intelligence*, pages 813–834, Chambéry, France, August 1993. Morgan Kaufmann Publishers.

[18] I. Kononenko. Successive naive Bayesian classifier. *Informatica*, 17:167–174, 1993.

[19] I. Kononenko and I. Bratko. Information-based evaluation criterion for classifier's performance. *Machine Learning*, 6:67–80, 1991.

[20] P. Kontkanen, P. Myllymäki, and H. Tirri. Probabilistic instance-based learning. Submitted to ICML'96, January 1996.

[21] R.J.A. Little and D.B. Rubin. *Statistical analysis with missing data.* Wiley, 1987.

[22] D. Michie, D.J. Spiegelhalter, and C.C. Taylor, editors. *Machine Learning, Neural and Statistical Classification.* Ellis Horwood, London, 1994.

[23] P. Myllymäki and H. Tirri. Bayesian case-based reasoning with neural networks. In *Proceedings of the IEEE International Conference on Neural Networks*, volume 1, pages 422–427, San Francisco, March 1993. IEEE, Piscataway, NJ.

[24] P. Myllymäki and H. Tirri. Massively parallel case-based reasoning with probabilistic similarity metrics. In S. Wess, K.-D. Althoff, and M Richter, editors, *Topics in Case-Based Reasoning*, volume 837 of *Lecture Notes in Artificial Intelligence*, pages 144–154. Springer-Verlag, 1994.

[25] P. Myllymäki and H. Tirri. Constructing computationally efficient Bayesian models via unsupervised clustering. In A.Gammerman, editor, *Probabilistic Reasoning and Bayesian Belief Networks*, pages 237–248. Alfred Waller Publishers, Suffolk, 1995.

[26] R.E. Neapolitan. *Probabilistic Reasoning in Expert Systems.* John Wiley & Sons, New York, NY, 1990.

[27] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann Publishers, San Mateo, CA, 1988.

[28] J. Rissanen. *Stochastic Complexity in Statistical Inquiry.* World Scientific Publishing Company, 1989.

[29] D. Roth. On the hardness of approximate reasoning. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, volume 1, pages 613–618, 1993.

[30] D.W. Scott. *Multivariate Density Estimation. Theory, Practice, and Visualization.* John Wiley & Sons, New York, 1992.

[31] D. Skalak. Prototype and feature selection by sampling and random mutation hill climbing algorithms. In *Machine Learning: Proceedings of the Eleventh International Conference*, pages 293–301, 1994.

[32] D. Titterington. Updating a diagnostic system using unconfirmed cases. *Applied Statistics*, 25:238–247, 1976.

[33] D.M. Titterington, A.F.M. Smith, and U.E. Makov. *Statistical Analysis of Finite Mixture Distributions.* John Wiley & Sons, New York, 1985.